Universidade Federal do Paraná

Departamento de Informática

Fábio Engel de Camargo
Elias P. Duarte Jr.

Difusão com Informações de Vizinhança em Redes MANET Aplicada para Manutenção da Topologia e Consenso

Relatório Técnico
RT_DINF 001/2021

Curitiba, PR
2021

Difusão com Informações de Vizinhança em Redes MANET Aplicada para Manutenção da Topologia e Consenso

Fábio Engel de Camargo
Universidade Federal do Paraná
Departamento de Informática
fabioengel@ufpr.br

Elias P. Duarte Jr.
Universidade Federal do Paraná
Departamento de Informática
elias@inf.ufpr.br

Resumo

Uma MANET (*Mobile Ad hoc NETwork)* é uma rede ponto-a-ponto em que um conjunto de dispositivos móveis sem fio cooperam para a transmissão de mensagens. Uma mensagem pode trafegar por diversos dispositivos intermediários até chegar ao destino. Neste trabalho apresentamos um algoritmo de difusão de melhor esforço para MANETs. O algoritmo é utilizado no contexto de um detector de falhas que permite que cada dispositivo mantenha a topologia da rede localmente. O algoritmo é utilizado por um dispositivo que detecta uma modificação da topologia para disseminar a informação para os demais. Como outro exemplo de aplicação que pode se beneficiar do detector, apresentamos sua aplicação em uma estratégia de consenso inspirada no algoritmo Raft. O algoritmo de difusão proposto utiliza uma estratégia baseada no conhecimento que os nodos da rede têm sobre vizinhos, sendo mais eficiente que a inundação, que gera grande número de mensagens redundantes. Uma versão preliminar do algoritmo é especificada e avaliada através de simulação, que considera uma composição estática de nodos.

1. Introdução

Uma rede MANET (*Mobile Ad hoc Network*) é composta por dispositivos que estabelecem comunicação sem-fio entre si, sem o auxílio de uma infraestrutura fixa [Magnus Frodigh and Larsson 2000]. Cada dispositivo membro é um nodo da rede que atua tanto como roteador e como *end-host*, ou seja, dispositivo de usuário final. Enquanto roteador, o nodo encaminha mensagens de dispositivos que encontram-se fora de alcance mútuo. Desta forma, o modelo de sistema distribuído que executa em uma rede MANET é ponto-a-ponto (*multi-hop*). As redes MANET são ditas dinâmicas, na medida em que os dispositivos são livres para mover-se, podendo decidir por entrar ou sair da rede em qualquer instante. Isto faz com que as topologias criadas sejam arbitrárias e variáveis ao longo do tempo.

Existem diversas estratégias para a realização de difusão (*broadcast*) em redes MANETs [Bakhouya 2013]. A abordagem mais simples é aquela baseada em inundação (*flooding*), em que cada mensagem recebida pela primeira vez é reencaminhada por todos os nodos da rede. Entretanto, esta abordagem faz com que um número grande de mensagens redundantes sejam transmitidas. Tendo em vista a tecnologia para encaminhando de mensagens, a inundação pode inclusive aumentar as colisões nos enlaces físicos sem-fio. Apresentamos neste trabalho um algoritmo para difusão de eventos em redes MANET

que reduz a quantidade de retransmissões realizadas em comparação com a inundação. O algoritmo de proposto é mais eficiente que a inundação pura, pois um nodo utiliza informações de vizinhança para evitar a retransmissão de mensagens desnecessárias.

O algoritmo é preliminar no sentido que assume uma rede de composição estática, baseada em um conjunto de n processos conectados em uma MANET de topologia arbitrária. Um processo corresponde a um dispositivo/nodo da rede. A rede pode se particionar e, mais tarde, as partições podem se unir novamente (*healing*). A difusão é de melhor esforço (*best-effort broadcast*) garantindo a entrega no componente conexo ao qual a origem pertence, desde que a origem não falhe. O algoritmo é especificado assumindo um sistema distribuído assíncrono ponto-a-ponto. Os processos estão sujeitos a falhas por parada e recuperação (*crash-recovery*) [Cachin et al. 2011]. Um processo mantém informações de estado em memória não volátil, de forma que quando recupera tem as informações armazenadas antes de falhar.

O algoritmo de difusão é especificado e apresentado tanto no contexto de uma estratégia de consenso distribuído inspirada no algoritmo Raft [Ongaro and Ousterhout 2014] como no contexto de um detector de falhas. O detector tem por objetivo permitir que os nodos da rede MANET mantenham localmente informações sobre a topologia da rede. Estas informações podem ser usadas para diversos própositos [Banzi et al. 2011], como roteamento, posicionamento e recuperação de serviços e dados, balanceamento de carga, entre outros. O algoritmo de difusão é usado para a disseminação de eventos que correspondem a alterações da topologia da rede. Na medida em que a rede é dinâmica, é um desafio garantir que as informações refletem a topologia real da rede. Os processos se monitoram trocando mensagens periódicas informando que estão corretos (*heartbeats*). Para acomodar visões conflitantes de dois processos sobre um vizinho em comum, são mantidas informações de estado dos arcos (arestas direcionadas) ao invés de sobre processos propriamente ditos. Desta forma, cada nodo que detecta um evento que corresponde a uma alteração da topologia – concretamente pode detectar que o enlace para um vizinho falhou ou recuperou – envia por difusão informações para que todos os demais nodos possam fazer as suas atualizações locais.

Tanto o detector de falhas baseado no algoritmo de disseminação de mensagens proposto, como sua aplicação para a execução do consenso foram implementadas através de simulação. Experimentos demonstram que o algoritmo proposto consegue reduzir substancialmente a quantidade de mensagens transmitidas em comparação à inundação.

O restante deste trabalho está organizado da seguinte maneira. A Seção 2 descreve o modelo de sistema, bem como o detector de falhas para redes MANET proposto e a aplicação exemplo inspirada no algoritmo Raft. Na seção 3 apresentamos a estratégia para difusão de mensagens. Na Seção 4 são descritas as simulações e os resultados obtidos. Por fim, a conclusão segue na Seção 5.

2. Modelo de Sistema, Detectores de Falhas e Consenso

Neste trabalho uma rede MANET é modelada como um sistema distribuído representado como um grafo $G = (\Pi, E)$, no qual o conjunto de vértices $\Pi = \{p_1, p_2, ..., p_n\}$ consiste de n processos, aos quais são assinalados identificadores sequenciais únicos, de 1 até n. O conjunto E de arestas corresponde à capacidade dos processos de comunicarem diretamente entre si, sem passar por intermediários. A topologia que conecta os processos

é arbitrária, ou seja G não é sempre representável por um grafo completo, desta forma $\forall p_i, pj \in \Pi : \exists (p_i, p_j) \notin E$.

Desta forma, para que dois processos sem uma aresta em comum possam se comunicar, é necessário que as mensagens transmitidas entre eles passem por intermediários. Este tipo de sistema é denominado ponto-a-ponto ou *multi-hop*. Assumimos que os processos conhecem seus vizinhos na rede.

O sistema é assíncrono, isto é, não há restrições temporais conhecidas nem sobre o tempo de transmissão de mensagens, nem sobre o tempo que os processos necessitam para completar tarefas. Os processos podem falhar por parada, podendo mais tarde se recuperar. Portanto modelo de falhas adotado é o *crash-recovery*. Cada processo tem memória não volátil na qual mantém informações de estado, de forma que, ao recuperar, o processo é capaz de carregar informações do estado antes da falha.

Detectores de Falhas

Neste trabalho apresentamos um algoritmo de difusão de melhor esforço para redes MANET. O algoritmo é utilizado no contexto de um detector de falhas, cujo modelo é especificado a seguir. Antes, para compreender o contexto em que os detectores foram originalmente propostos, é preciso mencionar o consenso. O consenso é o problema básico do acordo em sistemas distribuídos [Lamport 2001], que pode ser informalmente descrito da seguinte maneira. Os processos do sistema fazem inicialmente propostas de valores, dentro de um conjunto pré-definido de valores possíveis. Ao final do consenso, todos os processos decidem por um mesmo valor, que está entre os valores propostos. Em 1985, foi provado [Fischer et al. 1985] que o consenso é impossível em sistemas assíncronos sujeitos a falhas. A raiz desta impossibilidade é a dificuldade de diferenciar um nodo lento de um nodo falho.

Anos mais tarde [Chandra and Toueg 1996], os detectores de falhas foram propostos como uma abstração para contornar esta impossibilidade. Os detectores surgiram no seguinte contexto: se os processos do sistema distribuído tiverem acesso a informações sobre quais processos do sistema falharam, é possível realizar o consenso? Os autores classificaram os detectores de falhas de acordo com duas propriedades: completude e precisão. A completude determina que processos falhos são efetivamente detectados como tal. A precisão determina que processos corretos não são erroneamente suspeitos de terem falhado. A partir destas duas propriedades e variações, que levam em consideração que o detectores podem cumprir as propriedades após um tempo, oito classes de detectores são apresentadas e os autores mostram que mesmo a mais fraca é capaz de garantir a realização do consenso em sistemas distribuídos assíncronos sujeitos a falhas crash.

Entretanto, se um sistema executando o detector de falhas violar as propriedades do detector mais simples, o consenso não vai ser possível. A estratégia adotada pelos desenvolvedores de algoritmos de consenso é garantir a segurança (*safety*) do algoritmo, isto é, garantir que em nenhum caso os processos executando o algoritmo vão decidir por valores diferentes. Infelizmente, a impossibilidade do consenso se revela com o fato de que a propriedade de progressão (*liveness*) do algoritmo pode não ser cumprida, significando que este pode nunca terminar. Entre os algoritmos que seguem esta linha, destacamos o Paxos [Lamport 2001] e o Raft [Ongaro and Ousterhout 2014].

O detector de falhas para MANETs utilizado no trabalho estende a definição ori-

ginal, na medida em que permite que os nodos mantenham informações não apenas sobre os processos falhos do sistema, mas toda a topologia da rede. Desta forma, cada processo p_i mantém um grafo $G_i = (\Pi_i, E_i)$ que representa sua visão local da topologia da rede. Utilizando G_i, o processo p_i determina o componente conexo ao qual pertence, isto é, a porção da rede atingível em um determinado momento. Falhas de processos por parada (*crash*) podem ocorrer de forma inclusive a particionar a rede. Os processos monitoram seus vizinhos, trocando mensagens (*heartbeats*) informando que estão corretos. Assume-se que a completude do detector de falhas, pois um processo falho para de mandar *heartbeats* e será detectado após um tempo. Entretanto como o sistema é assíncrono, as falsas suspeitas são inevitáveis.

No modelo proposto, é preciso que o detector de falhas acomode situações em que dois vizinhos de um mesmo processo têm visões conflitantes sobre o estado do mesmo. Concretamente, a situação é a seguinte: um dos vizinhos considera o processo correto, tendo recebido mensagens corretamente enquanto o outro tem uma falsa suspeita. A Figura 1 ilustra esta situação, na figura o processo p_b tem dois vizinhos, p_a e p_c. Enquanto p_a detecta que p_b está correto, p_c suspeita incorretamente que p_b está falho. Para acomodar esta situação, o detector mantém informações não sobre o estado de processos propriamente ditos, mas dos arcos direcionados (p_o, p_d) que conectam um processo origem (p_o) a um processo destino (p_d). Desta forma, um detector D é um conjunto de duplas $(arco, estado)$. Cada aresta de (p_i, p_j) de E está representada pelos dois arcos direcionados correspondes: (p_i, p_j) e (p_j, p_i), cada um indicando a visão que o nodo origem tem do nodo destino - o estado pode ser *correto* ou *suspeito*.

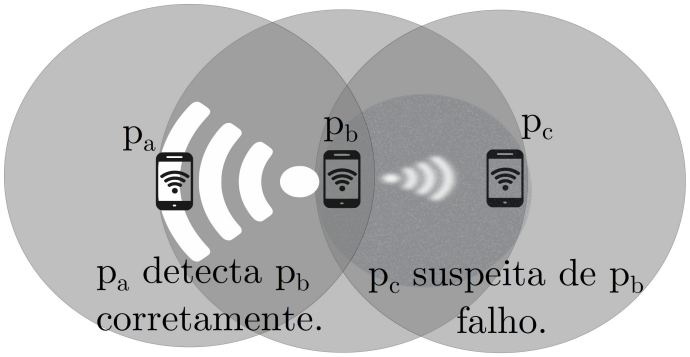


Figura 1. Dois vizinhos de um mesmo nodo tem visões diferentes sobre seu estado.

Aplicação Exemplo: Consenso

Uma estratégia de consenso é usada neste trabalho como exemplo de aplicação que utiliza o algoritmo de difusão de mensagens em MANET bem como o serviço de detecção de falhas que mantém a topologia da rede localmente em cada nodo. A estratégia de consenso implementada na simulação é baseada no algoritmo Raft [Ongaro and Ousterhout 2014]. Um processo pode assumir um de três estados possíveis durante a execução do algoritmo: líder (*leader*), seguidor (*follower*) e candidato (*candidate*). O líder é o responsável por propor valores, estas proposições partem apenas dele em direção aos demais processos. A sua existência é reconhecida por meio da recepção de mensagens do tipo *Leader-heartbeat*, isto é, mensagens periódicas transmitidas em intervalos constantes para todos indicando sua existência.

Todos os processos iniciam no estado seguidor. Se por um intervalo de tempo pré-definido o seguidor não receber uma mensagem de Líder *Leader-heartbeat*, ele assume sua ausência e imediatamente ativa um intervalo de tempo limite, chamado aqui de *candidateTimeout*. Se ao final deste intervalo de tempo não for recebida nenhuma mensagem de *Leader-heartbeat* ou pedido de voto (explicado a seguir), o processo altera seu estado para candidato. Para que seja pouco provável que todos, ou mesmo muitos, assumam o estado de candidato no mesmo instante de tempo, o algoritmo faz com que o valor de *candidateTimeout* seja aleatório dentro de um limite pré-definido, com o objetivo portanto, de ser adotado um *timeout* diferente para cada processo.

Todos os processos iniciam no estado seguidor. Se por um intervalo de tempo pré-definido o seguidor não receber uma mensagem *Leader-heartbeat*, ele assume que não há líder e imediatamente ativa um intervalo de tempo limite, chamado aqui de *candidateTimeout*. Se ao final deste intervalo de tempo não for recebida nenhuma mensagem *Leader-heartbeat* ou Pedido de Voto (explicado a seguir), o processo altera seu estado para candidato. Para que seja pouco provável que todos, ou mesmo muitos, assumam o estado de candidato no mesmo instante de tempo, o algoritmo faz com que o valor de *candidateTimeout* seja aleatório dentro de um limite pré-definido, com o objetivo portanto, de ser adotado um *timeout* diferente para cada processo.

Ao tornar-se candidato, o processo dá início ao mecanismo de eleição de líder, em um novo termo. Este mecanismo inicia com o candidato votando em si próprio, em seguida transmite uma mensagem com um Pedido de Voto. Esta mensagem é encaminhada para todos os processos da rede. Ao receber esta mensagem, um processos seguidor verifica se a ausência de líder é confirmada e se não votou neste termo. Em caso afirmativo, o processo imediatamente envia uma mensagem confirmando seu voto até o candidato.

Uma vez que o candidato receba uma maioria de votos, ele altera seu estado para líder, em seguida inicia o envio periódico de mensagens *Leader-heartbeat* para todos os demais processos

Havendo um líder, este pode propor valores, normalmente vindos de um cliente. As mensagens de Proposição são encaminhadas aos demais processos. Ao receber uma mensagem do tipo Proposição, um processo pode concordar e enviar uma mensagem de Resposta ao líder. Quando o líder recebe mensagens do tipo Resposta de uma maioria dos processos, ele decide e envia para todos os processos uma mensagem do tipo Commit, indicando que todos podem realizar decidir pelo valor.

Destaca-se que a estratégia funciona corretamente na presença de partições na rede. Apenas é possível o consenso se houver um componente conexo com a maioria dos processos isto é, mais de $n/2$ processos.

3. A Estratégia de Difusão de Eventos em Redes MANET

No detector de falhas especificado na seção anterior, durante o monitoramento, um processo pode detectar a ocorrência de um *evento*, que neste trabalho é definido como uma alteração no estado de um arco para um vizinho. O estado pode ter mudado tanto de suspeito para correto, como vice-versa. Após a detecção da alteração, a informação deve ser difundida na rede de forma que todos os processos possam alterar suas visões locais da topologia. A estratégia de consenso apresentada baseada no algoritmo Raft também

faz uso de difusão para transmitir mensagens de diversos tipos para todos os nodos da rede (ou componente conexo, já que a rede pode estar particionada). Apesar de que neste trabalho a disseminação de eventos surge nestes dois contextos, a necessidade de transmitir informação de tal forma pode surgir em contextos diversos, como por exemplo para disseminar um alarme em uma situação em emergência.

A forma mais comum de disseminação de eventos em MANETs é a conhecida por *flooding* (inundação) [Kahn et al. 2009]. O *flooding* neste contexto consiste no encaminhamento de pelo menos uma vez de uma mensagem em *broadcast* por cada membro do sistema. Embora teoricamente possa garantir que todos recebam a mensagem, tal procedimento pode causar um excesso de mensagens, com múltiplas cópias redundantes sendo transmitidas pelos diversos processos da rede. Além disso problemas no meio físico podem ocorrer, pois a comunicação ocorre por ondas de rádio frequência, as múltiplas transmissões de mensagens podem causar contenção e colisões nas transmissões. Este problema é onhecido na literatura como *broadcast storm problem* [Ni et al. 1999]. Convém então a utilização de abordagens mais eficientes.

Os algoritmos de difusão (*broadcast*) em redes MANET podem ser organizados em quatro categorias [Mkwawa and Kouvatsos 2011]: (1) simples, (2) probabilísticos, (3) baseados em área e (4) baseados em conhecimento sobre os vizinhos. Cada uma destas categorias é descrita a seguir. (1) Na difusão simples (*Simple Flooding*) toda mensagem de broadcast enviada por uma origem que é um dispositivo conectado à rede é retransmitida uma única vez por cada um dos demais conectados. (2) Na difusão probabilística, cada dispositivo apenas retransmite baseado em uma probabilidade computada. (3) Os algoritmos baseados em área tiram partido do posicionamento dos processos na rede para melhorar a eficiência da inundação. E finalmente, (4) nos algoritmos baseados em conhecimento sobre vizinhos as decisões sobre o encaminhamento de uma mensagem enviada por difusão são tomadas baseadas no conhecimento sobre os vizinhos situados dentro de um limite de k-hops, isto é, vizinhos alcançáveis por meio de k saltos, sendo k um número inteiro de pequeno valor.

As propriedades desejáveis de um algoritmo de difusão em MANETs de acordo com [Nikolov and Haas 2015] são as seguintes:

(i) Uma mensagem transmitida por um processo da rede deve ser entregue a todos os demais processos.
(ii) A mensagem deve ser retransmitida o menor número de vezes possível, ou, de maneira equivalente, o número de vezes que uma mesma mensagem é recebida por um membro da rede deve ser minimizado.
(iii) Os processos da rede possuem apenas informações locais, ou seja, a figura de um controlador global é inexistente. Informações locais incluem, por exemplo, conhecimento sobre vizinhos ao seu alcance.
(iv) Mudanças na topologia durante a propagação de mensagens não devem afetar expressivamente as propriedade (i) e (ii).

Este trabalho descreve um algoritmo de difusão em redes MANET da categoria de algoritmos baseados em conhecimento sobre vizinhos. O algoritmo é dito de melhor esforço (*best-effort broadcast*) no seguinte sentido. A propriedade (i) acima só é cumprida se a origem permanecer sem falhas: apenas neste caso há a garantia que uma mensagem transmitida pela origem seja entregue por todos os processos. O problema que pode de-

correr da origem falhar antes de transmitir a mensagem para todos os seus vizinhos.

A seguir são descritos os passos do algoritmo proposto. Ao iniciar, o processo transmite uma mensagem de *heartbeat* para informar à todos ao seu alcance a sua existência. O processo que recebe esta mensagem pela primeira vez de um determinado processo, deve respondê-la em seguida com outra mensagem *heartbeat*. Periodicamente, todos os processos enviam mensagens de *heartbeat*. Deste modo, todo processo pode manter uma lista de quais processos estão ao seu alcance, isto é, seus vizinhos.

Quando deseja fazer *broadcast* de um novo evento, o processo insere na mensagem um identificador único para cada vizinho. Na implementação, pode ser utilizado, por exemplo, o endereço MAC da interface sem fio, sendo necessários, portanto, 6 octetos para cada vizinho.

Toda vez que um processo recebe um evento, ele verifica se este evento já foi recebido anteriormente. A decisão de encaminhamento ocorre apenas na condição de tratar-se de um novo evento, evitando assim retransmissões desnecessárias. Ao receber um evento considerado novo, o processo receptor compara a lista de processos recebida com a sua própria lista. Se sua lista contém os identificadores de todos os processos contidos na mensagem, isto é, todos processos ao alcance do emissor estão também ao alcance do receptor, a mensagem não é retransmitida. Caso contrário, a retransmissão da mensagem é feita, de modo a alcançar processos diferentes. A Figura 2 ilustra um exemplo.

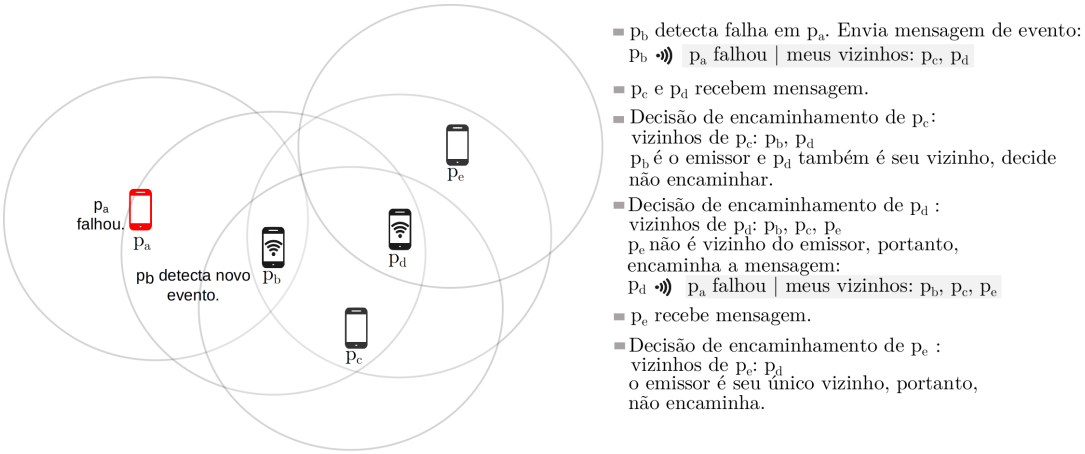


Figura 2. Exemplo de execução do algoritmo.

O algoritmo é apresentado em pseudo-código abaixo, executado pelo processo p_i. A primitiva `broadcast(m)` é usada pela origem para disparar a difusão da mensagem `m` em toda a rede. A primitiva `send(m,L)` é usada para um processo enviar a outro a mensagem +m+ e sua lista de vizinhos `L`. Assume-se um canal perfeito, ou seja a mensagem transmitida utilizando esta primitiva `send(m,L)` é entregue corretamente ao destino, caso origem e destino se considerem mutuamente corretos. A primitiva `receive(m,L)` corresponde ao recebimento da mensagem transmitida com a primitiva `send(m,L)`. Por fim, a primitiva `deliver(m)` corresponde a entrega da mensagem `m`.

```
Algorithm Best-Effort Broadcast Based on Neighborhood for MANETs
  // executed by process pi
  INIT: SentMessages = empty
  UPON broadcast(m):  send(m,Li)
                      SentMessages = SentMessages U {m}
                      deliver(m)
  UPON receive(m,Lj): if m not in SentMessages
                      then if the intersection of Li and Lj is not empty
                          then send(m,Li)
                               SentMessages = SentMessages U {m}
                               deliver(m)
```

O conjunto `SentMessages` mantém todas as mensagens transmitidas até o momento. Na implementação do algoritmo no contexto do detector de falhas este conjunto é mantido implicitamente: para confirmar que uma mensagem já foi recebida e retransmitida basta que o processo verifique se o evento que descreve está refletido na topologia mantida localmente.

Em uma rede MANET o canal de comunicação físico é *fair-loss*, desta forma a implementação da primitiva `send(m,L)` deve ser feita considerando confirmações e retransmissões, que são repetidas até que a confirmação chegue ou o destinatário é considerado falho. Desta forma, se a origem falha antes das mensagens serem efetivamente confirmadas pelos vizinhos, a difusão não garante a entrega na rede. Por este motivo o algoritmo é uma difusão de melhor esforço, a entrega só é garantida se todos a origem permanece correta.

Na próxima seção, o desempenho deste algoritmo é avaliado em comparação a estratégia de *flooding*. Além disso é também avaliado o detector de falhas que permite que os processos da rede MANET mantenham a topologia localmente e, como exemplo de aplicação, uma estratégia de consenso inspirada no algoritmo Raft.

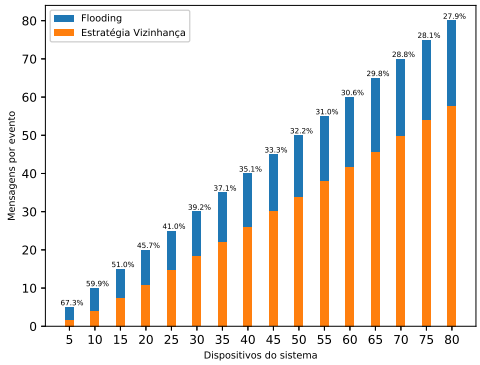
4. Simulação

O algoritmo de difusão proposto foi implementado atravé de simulação utilizando o OMNeT++ [OMNeT 2020]. O OMNeT++ é um simulador de eventos discretos, baseado em C++. Dois experimentos são reportados. O primeiro teve como objetivo avaliar a redução obtida na quantidade de mensagens transmitidas utilizando o algoritmo proposto para difusão baseado em conhecimento sobre vizinhos, utilizado para disseminar eventos sobre a topologia da rede. O segundo experimento foi executado para avaliar a quantidade de mensagens transmitidas pela estratégia de consenso baseada no algoritmo Raft.

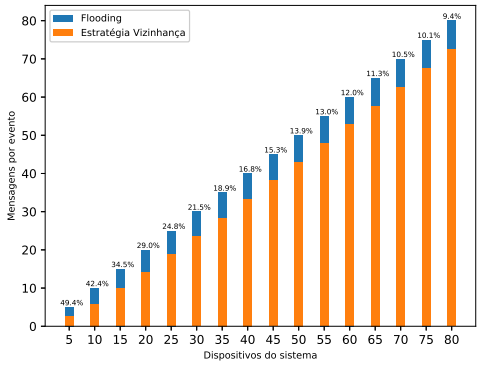
Para ambos os experimentos realizados, considerou-se uma distribuição aleatória dos dispositivos na área. Para todo dispositivo foi considerada a existência de uma inter-

face de comunicação sem fio com alcance de 80m. Considerou-se ainda, que dispositivos dentro desta área de alcance sempre recebem corretamente as informações transmitidas.

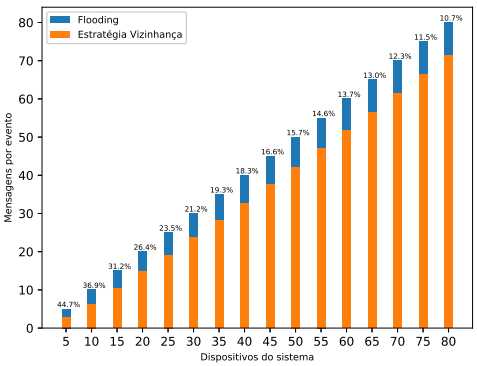
O primeiro experimento avaliou apenas o envio aleatório de eventos criados por todos os dispositivos do sistema. Três dimensões de cenários foram considerados, eles são representados pelas Figuras 3.a, 3.b e 3.c. Estas figuras apresentam a média de mensagens transmitidas ou encaminhadas para cada evento realizado. Observa-se que a Estratégia Vizinhança, isto é, a difusão proposta baseada em conhecimento sobre vizinhos, obtém uma redução substancial quando comparada à inundação (Flooding). Ao indicar a não retransmissão em situações as quais implicaria apenas em mensagens redundantes, a estratégia utilizada consegue uma redução em torno de 30% em cenários com uma grande concentração de dispositivos. Entretanto, a redução na quantidade de mensagens cai quando diminui a densidade da rede, como apontando pela Figura 3.c a qual tem dimensão de 400x400 metros.



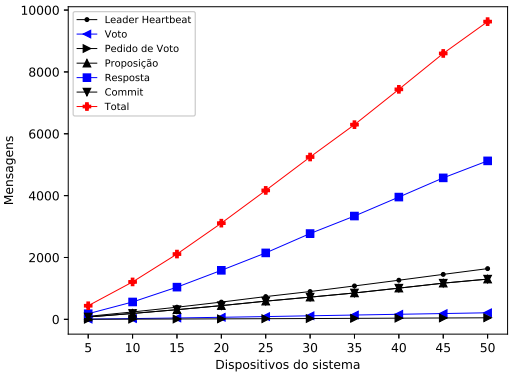
(a) Área 100m x 100m



(b) Área 200m x 200m



(c) Área 400m x 400m



(d) Algoritmo de Consenso

Figura 3. Simulações

O segundo experimento foi realizado para avaliar o algoritmo de consenso inspirado no Raft utilizando o algoritmo proposto para o envio de mensagens que exigem recebimento por todos da rede. O algoritmo utiliza mensagens dos seguintes tipos: *Leader-heartbeat*, Pedido de Voto, Proposição, Resposta e Commit. As mensagens de Voto e Res-

posta não exigem que todos tenham conhecimento, são mensagens direcionadas apenas ao líder. Portanto, não necessitam de difusão, mas ainda assim precisam ser roteadas pelos dispositivos da MANET. Para isto, foi necessário inserir novos campos nas mensagens de Pedido de Voto e Proposta. O emissor e todo dispositivo responsável pelo encaminhamento destas duas mensagens adiciona seu identificar no campo destinado a este fim, de modo que a mensagem ao chegar ao destino possua os identificadores dos dispositivos os quais a mensagem percorreu desde sua origem. Desta forma, é possível fazer com que as mensagens de Voto e Resposta sejam encaminhadas pelos mesmos dispositivos, sem necessidade de encaminhamentos desnecessários.

A Figura 3.d exibe a média de mensagens enviadas/encaminhadas obtida a partir da simulação de 1000 cenários distintos para cada quantidade de usuários no sistema, no qual cada cenário corresponde a uma distribuição diferente do posicionamento dos dispositivos. As mensagens que fazem parte do processo de eleição apresentam baixa quantidade, pois nas simulações foi eleito um único líder. Entretanto, fica evidente a grande quantidade de mensagens do tipo Resposta. Isto porque mensagens deste tipo são criadas por todos os dispositivos no estado seguidor em resposta a mensagem de Proposta enviada para o líder. O seu encaminhamento gera um alto tráfego na rede.

5. Conclusão

Este trabalho apresenta uma estratégia de difusão de melhor esforço baseda disseminação de mensagens com conhecimento sobre vizinhos para redes MANET. O algoritmo de difusão é aplicado para a disseminação de eventos sobre a topologia da rede em um detector que falhas que permite que os dispositivos da rede mantenham informações sobre a topologia. O algoritmo foi descrito, especificado e implementado através de simulação. Resultados obtidos demonstram que a redução na quantidade de mensagens transmitidas em relação com a inundação *flooding* depende da densidade da rede. Quanto mais densa, maior é a redução na quantidade de mensagens total transmitidas. Como exemplo de aplicação, o algoritmo de difusão e o serviço de detecção de falhas foram utilizados para implementar uma estratégia de consenso inspirada no algoritmo Raft. Na avaliação por simulação apresentou o número médio de mensagens transmitidas, de acordo com seu tipo. Fica evidente a importância de reduzir a quantidade de mensagens transmitidas por difusão.

Trabalhos futuros incluem permitir que a rede MANET seja efetivamente dinâmica, permitindo não apenas a saída, mas também a entrada de novos nodos, além de mobilidade.

Referências

Bakhouya, M. (2013). Broadcasting approaches for mobile ad hoc networks. In *2013 International Conference on High Performance Computing Simulation (HPCS)*, pages 705–707.

Banzi, A. S., Pozo, A. T. R., and Jr., E. P. D. (2011). An approach based on swarm intelligence for event dissemination in dynamic networks. In *30th IEEE Symposium on Reliable Distributed Systems (SRDS 2011), Madrid, Spain, October 4-7, 2011*, pages 121–126. IEEE Computer Society.

Cachin, C., Guerraoui, R., and Rodrigues, L. (2011). *Introduction to Reliable and Secure Distributed Programming*. Springer Publishing Company, Incorporated, 2nd edition.

Chandra, T. D. and Toueg, S. (1996). Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382.

Kahn, I. A., Peng, E., and Qian, H. L. (2009). An improved cover angle-based broadcasting algorithm for manets. In *2009 International Conference on Future Computer and Communication*, pages 241–245.

Lamport, L. (2001). Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58.

Magnus Frodigh, P. J. and Larsson, P. (2000). *Wireless ad hoc networking—The art of networking without a network*, pages 248–263. Number 4. Ericsson Review.

Mkwawa, I.-H. M. and Kouvatsos, D. D. (2011). *Broadcasting Methods in MANETS: An Overview*, pages 767–783. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S., and Sheu, J.-P. (1999). The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, MobiCom '99, page 151–162, New York, NY, USA. Association for Computing Machinery.

Nikolov, M. and Haas, Z. J. (2015). Towards optimal broadcast in wireless networks. *IEEE Transactions on Mobile Computing*, 14(7):1530–1544.

OMNeT (2020). Omnet++ discrete event simulator.

Ongaro, D. and Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 305–319, Philadelphia, PA. USENIX Association.